Renewable and Sustainable Energy Reviews 27 (2013) 401–406

Contents lists available at SciVerse ScienceDirect

Renewable and Sustainable Energy Reviews

journal homepage: www.elsevier.com/locate/rser

Plug-in driven architecture for renewable energy generation monitoring

F. Xavier Villasevil *, Julio E. Vigara, Lautaro Chiarle [1]

Universitat Politècnica de Catalunya, Spain

A R T I C L E I N F O

Article history:
Received 29 October 2012
Received in revised form
24 June 2013
Accepted 30 June 2013
Available online 31 July 2013

Keywords:
Plug-in
Monitoring
Renewable energy

A B S T R A C T

This paper is about the benefits of a plug-in based software architecture, but it is not only intended for programmers. It also stands the fact that monitoring is needed in any renewable energy generation plant and describes a way to fulfill this need.

The fast evolution of renewable energy sources during the last decades resulted in the installation of many power systems all over the world, showing a sharp trend towards Distributed Resources (DR). There are many people related to them (because they use, own, operate, maintain or receive services from DR) who may be interested in knowing things about them for several reasons like production, maintenance, investigation, profitability, etc.; that is why data-acquisition systems are widely used in renewable energy source applications. These systems allow to collect and analyze data regarding the renewable energy installation performance as well as meteorological data.

The basic idea on this paper is to achieve monitoring of different energy generation plants using a plug-in driven design technique, presenting as a result of an extensible software architecture design.

The main difference between this paper and the previous work is that we focus specifically on monitoring as a necessity, proposing a generic and extensible software system, which could be employed to monitor any kind of renewable energy plant and extended to connect to new developed devices by changing small pieces of software.

Contents

1. Introduction

Renewable energy is the name given to the energy obtained from natural sources virtually inexhaustible, either by the vast amount of energy they contain, or because they are able to be regenerated by natural means. We can name, for instance, solar power, wind power, geothermal power, fuel cells, etc. [1].

* Corresponding author. Tel.: +34 938 967 728.
E-mail addresses: villasevil@eel.upc.edu.(F.X. Villasevil) julio.vigara@upc.edu.(J.E. Vigara)chiarlelautaro@gmail.com.(L. Chiarle)
[1] Universitat Politècnica de Catalunya INSIDE research group collaborator.

1364-0321/$ - see front matter
http://dx.doi.org/10.1016/j.rser.2013.06.048

For years and years, fossil fuels have been enormously consumed on the Earth [2], so they have been rapidly decreased, making it more difficult and expensive to use them. In addition, the pollution produced by fossil fuels made people realize the importance of renewable energy. During the last years, many people have dedicated to the research, development and utilization of such important energy, causing the need to monitor its generation.

The following Section enumerates some existing technical solution purposes on that matter. Section 3 describes the main characteristics of a plug-in architecture and its benefits, justifying its use.

Section 4 goes on detailing our proposed solution, an architecture which establishes the basis for collecting data from a renewable energy generation plant, visualizing them both in a statistical as a real-time manner and notifying the user about several events and anomalies that may occur.

Finally, Section 5 reveals the conclusions reached and Section 6 points out some ideas that can contribute to future works.

2. State of the art

In this section we present part of the current work on renewable energy generation monitoring. Several authors have been investigating on this subject and achieved different goals.

While some of them focus on monitoring the installation state and controlling involved devices, others prefer to pay attention to data related with the installation performance and meteorological behavior. Among important aspects usually monitored we must mention connection status, real power, reactive power, voltage, frequency and energy (kWh). In some of these articles alarm and notification issues are also treated.

du Boulay et al. [3] show that it is possible not only to monitor but also to control sophisticated instruments remotely. In their article they present the web services based system for collaborative monitoring of remote scientific instruments and sensors.

Kuo-Hua Liu [1] studied the dynamic characteristics of photovoltaic energy generation using a PLC control. The present work includes real-time voltage and current monitoring, event detection, information collection and the set-up of a security system.

There are also various papers which focuses on data collection, Koutroulis et al. [4] as well as N. Forero et al. [5] introduce interesting articles about data acquisition. On the first one, the proposed system consists of a set of sensors for measuring both meteorological and electrical parameters. The collected data is then sent to a LabVIEW program, which is used to further process, display and store it in the PC disk. The second one's aim is to present a system developed for monitoring PV solar plants using a novel procedure based on virtual instrumentation. The measurements and processing of data are made using high precision I/O modular field point devices as hardware, a data acquisition card as software and the package of graphic programming, LabVIEW.

Benghanem and Maafi [6] go beyond, developing an expert system for studying the performance of photovoltaic (PV) systems; in particular the sizing of such systems and their reliability. They show it is possible to exploit the information about the climatic environment of a given site and the parameters related to PV systems. On this work data are treated and decoded in their real physical values (current, voltage, energy, etc.) and used by a PC computer to study the performance of the PV system considered.

On the other hand, papers like [7] emphasize on the analysis of modules performance, degradation or failure under realistic outdoor conditions, showing another important responsibility of monitoring systems: failure detection.

The inclusion of DG (Distributed Generation) systems is changing the paradigm of energy production. Local Grid Codes dictate the behavior of the grids under different faults. However, in grids with high penetration of renewable energy sources, conventional regulations are under revision. The basic element for interconnecting DG to the transmission system is the three phase inverter. In normal grid conditions, three phase DG inverters inject all the generated active power into the grid, but one of the major drawbacks for proper operation of the whole installation occurs when a voltage sag is transmitted through the network. Depending on the depth and duration of the voltage sag, Grid Codes may force even a temporary disconnection of the system.

A flexible voltage support control scheme has been proposed in [8] for three phase DG inverters under grid fault. In grid fault conditions, the controller must react to the perturbation and mitigate the adverse effects on the inverter side. The purposed voltage support strategy can be modified by means of a control parameter according to the type of voltage sag, resulting in a flexible combination of raising and equalizing capabilities. Voltage sags and other significant issues that may occur will be easier to understand as more information we have about the behavior of DG, therefore monitoring is an extremely useful tool as far as research is concerned.

Finally, Li Wang and Kuo-Hua Liu [2] show that it is as important to know the state of the system at a given time as to be notified when something is not going as expected. Their system performs different functions such as power-flow data measurement and collection, realtime load monitoring and load-shedding control, real-time and historical data mapping, timely report and handle of alarms and events.

All of these papers show us how important monitoring is and allow us to identify main parts in any monitoring system, such as: data collection, data processing and visualization and finally, notification and handle of alarms and events.

3. About plug-ins

A plugin is just like a little black box that provides a public interface to the outside, which can be used by other plugins. In plug-ins based software systems the main application logic is reduced to mere plugins hosting and management, as it provides no other functionality to users [9]. The real feature set offered by these applications is defined by different individuals plugins loaded and managed by this base infrastructure. Communication and interaction between plugins is achieved by using its public interfaces.

A plug-in can also provide general functionality allowing other plug-ins to extend or customize portions of it through an Extension Point. The extension point declares a contract that extensions must conform to. Plug-ins with more specific functionality may connect to that extension point by implementing that contract. The key attribute is: the plug-in being extended knows nothing about the plug-in that is connecting to it beyond the scope of that extension point contract. This allows plug-ins built by different individuals or companies to interact seamlessly, even without them knowing much about one another.

By dividing an application into multiple plugins, giving each one a well-defined task, complexity and size of each module is drastically reduced [9]. Also, when compiling a plugin into a well-documented executable unit, the plugin becomes a sealed unit with a clearly defined interface and can be shared between various developers, without having to take a look at the original source code.

A high degree of modularity is an important factor for building software systems of high quality [9]; however, proliferation of the functionality of a plugin-based system could be detrimental to its ease of use and bring to confusion and disorder if users who have

no prior knowledge of the available functions and how to access them are unable to find those that suit their needs [10].

Anyway, when getting a fine-grained modularity, not only in the source code, but also in the post-build level, plugin frameworks help manage complexity, simplify configuration and deployment of applications and allow users or third parties to improve existing applications easily developing their own modules without having to access the full source code [11].

Not only because of the source code split into different modules, but for being able to compile these modules in closed blocks ready for use in software construction. Thus, the development of an application in whole or a complex software system is reduced to the task of selecting, combining and distributing the appropriate modules. In conclusion, plugin-based software systems have become very popular, as they are the next step in the evolution of application development [11].

3.1. Benefits

Apart from the typical improvements of testability, reusability, readability, and maintainability that are already supported by classical modularization techniques such as object-oriented modeling, plugin-based software development provides the following advantages [11,9]:

- Modularity is an important abstraction mechanism, not only at source code level but also at application level. It helps to handle the high degree of complexity we have to deal with today when facing large software systems.
- Developers do not need to be distracted by reviewing other's source code, as plugins are equipped with a comprehensive and yet easy to use application programming interface (API). Furthermore, development of drivers and stubs to test each module independently is also simplified.
- It is much easier to customize software systems to meet specific requirements in a customer's scenario just by adding necessary and removing irrelevant parts.
- Application deployment is simplified. To deploy new functionality or to update outdated plugins from update locations (i.e, servers) through a network is easily achievable.
- In addition, this architecture gives third parties the opportunity to develop extensions to an existing application independently.

Aside these benefits, there are some disadvantages that could be useful to highlight:

- Complexity does not disappear entirely [9]. In fact, it is transferred from the programming level to modeling level, since final application has to be assembled by the combination of different small and simple plugins. Then comes the issue of glue code, which is required to link and combine these modules.
- To handle incompatibilities between different plugins versions is a very important matter to be had into account. When changing a plug-in version, compatibility must be checked with the rest of plugins forming the application, in order to avoid inconsistencies.

3.2. Using plug-ins on energy generation monitoring

The mentioned problem of “glueing” the pieces can be minimized by applying a well-defined subsystems division based on the application domain, in our case the energy generation monitoring.

The natural extensibility offered by plug-in applications will make it easy to add new connections between the monitoring system and the generation plant when a new Distributed Resource [12] is added. For instance, the application could be extended with connections to a new brand of solar panels by changing a simple plugin.

Moreover, customized versions of the application for different kinds of energy generation plants can be built by adding and/or removing existing plug-ins or by changing a particular plug-in version; always considering that it is crucial to pay attention to plug-in versions compatibility.

Using plug-ins, a complex problem is divided into smaller and less complex problems, which are solved by this small software components. Each plug-in represents independent functionality and can be treated that way. This “divide and conquer” strategy fits perfectly in a complex domain as the energy generation monitoring.

4. Purposed monitoring architecture

A renewable energy plant monitoring system (Fig. 1) must permit to know the installation state and notify in a quick and efficient manner about any anomaly that may occur. It can become the best tool for the plant energy management. Basically, it has two main goals:

- Watch the production matches the expected, and is in accordance with the forecast.
- When production is not as awaited, it must detect responsibilities and possible solutions. There are 3 basic functional points identified in any monitoring system: data acquisition, data visualizing and notification. The subsystems are crossed transversally by a data model.

This article focuses on data acquisition subsystem, which was built based on an extensible software architecture in order to communicate with several DG systems.

4.1. Data acquisition

According to IEEE 1547 Series of Standards [12] Section 5, “Each DR (Distributed Resources) unit of 250 kVA or more or DR aggregate of 250 kVA or more at a single PCC (Point of Common Coupling) shall have provisions for monitoring its connection status, real power output, reactive power output, and voltage at the point of DR connection”. Those MIC (monitoring, information exchange, and control) provisions will be collected by the Data Acquisition subsystem. The Data Acquisition subsystem operates isolated from the data visualizing and notification modules. It is responsible for communicating with the different components that constitute the power plant, receiving information from them in real time. Within this subsystem there may be several components (or objects) extending the “Receiver” class. Each specific Receiver will be responsible for meeting the requirements established by the IEA (Information Exchange Agreement) according to the IEEE 1547 Series of Standards [12] and contributing to information security maintenance (confidentiality, integrity and availability). A Receiver responds to the “receive()” message (sent by an Scheduler) returning a collection of events (Event class). An event can be: – A failure (Failure), represented by an error code and a parameters list. For example a connection failure or an internal failure of the plant (an inverter reporting that a panel is not working properly) – A measure (Measure), represented by a magnitude, a unit and a component which originated it. For example, the voltage in volts of a particular inverter.

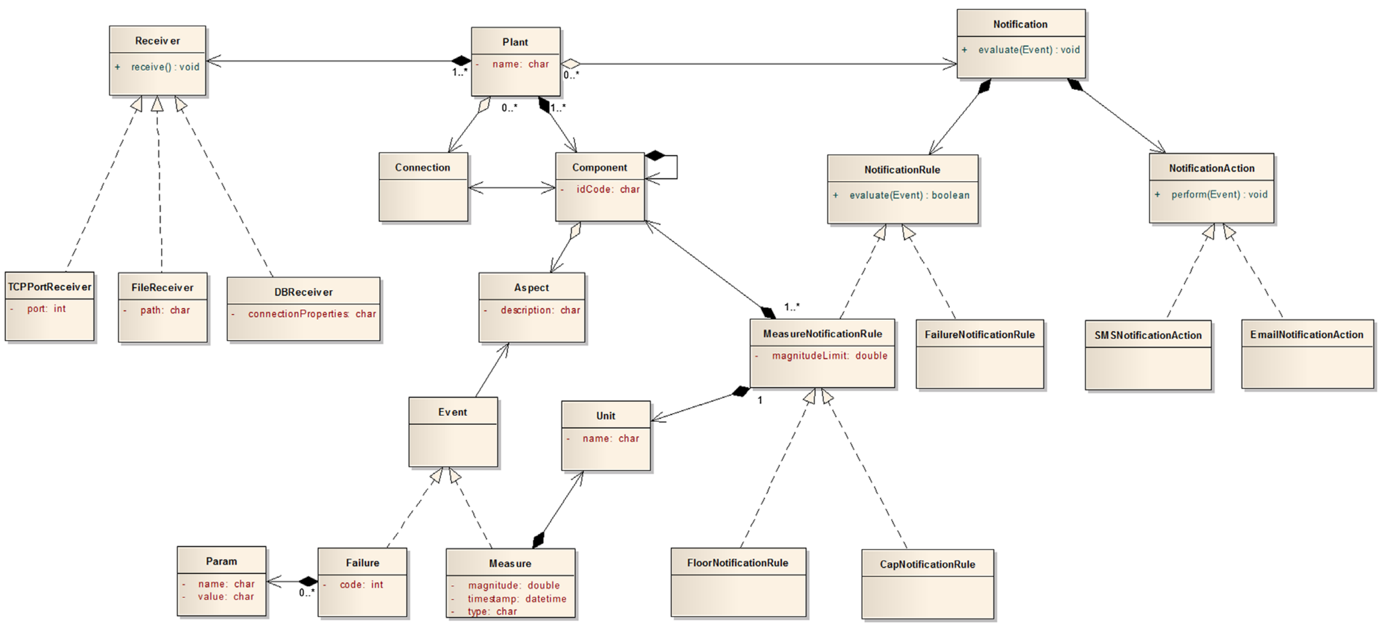


Fig. 1. Generic monitoring model.

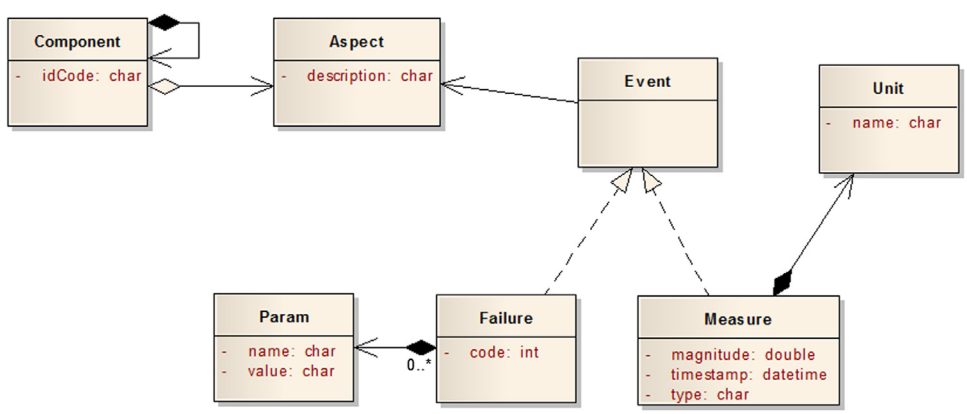


Fig. 2. Event object model.

The most relevant part of this design is not the Receiver itself but the Event object model (Fig. 2), which is used by the application as a contract to communicate with any Receiver. It suits the requirements stated by any IEE 1547 Information Exchange Agreement, as attributes for each of the classes in the ontology can be mapped into an Aspect on the EventModel, classes in the ontology can be mapped to Component class extensions and units and magnitudes are known values. This model allow us to implement the use cases identifying the information exchange needs for accomplishing real-world interactions. Basing communication on this contract any kind of Receiver could be implemented, i.e. to communicate with an Inverter by an RJ45 connection or through a database. Thus, it is possible to implement a Receiver for measuring any element or even a manufacturer could provide their own Receiver, providing its measurements based on this design. Some Receivers implemented by way of example during this study are listed below:

- OmronPLCReceiver: it communicates with a PLC Omron and returns the voltage and current measurements from it (as a collection with two Measures from the PLC, one with Volt unit and one with Ampere unit). As it is an example this implementation is very simple and does not return failures.
- XMLReceiver: as the Event object model, like any other, can be serialized into XML language (specified by the W3C [13]) we have implemented a Receiver capable of reading XML files to build a collection of Events. For example, a measure of voltage and current in XML based on this design would be something like:

<measure aspect='VoltageRating' unit='Volt' magnitude='12.3' component='Inverter X'/>

<measure aspect='AmpRating' unit='Amper' magnitude='9.8' component='Inverter X'/>

4.2. Extension

To build a specific generation plant monitoring system, for instance to monitor a photovoltaic plant, it is necessary to plug model extensions to manage the photovoltaic generation plant data. Typically those extensions will be based on IEEE 1547 Series of Standards [12] Information Exchange Model. Fig. 3 presents some of the elements needed to be modeled in order to extend the model presented in Fig. 1, such as: solar plant, solar panels, inverters, light sensors, etc.

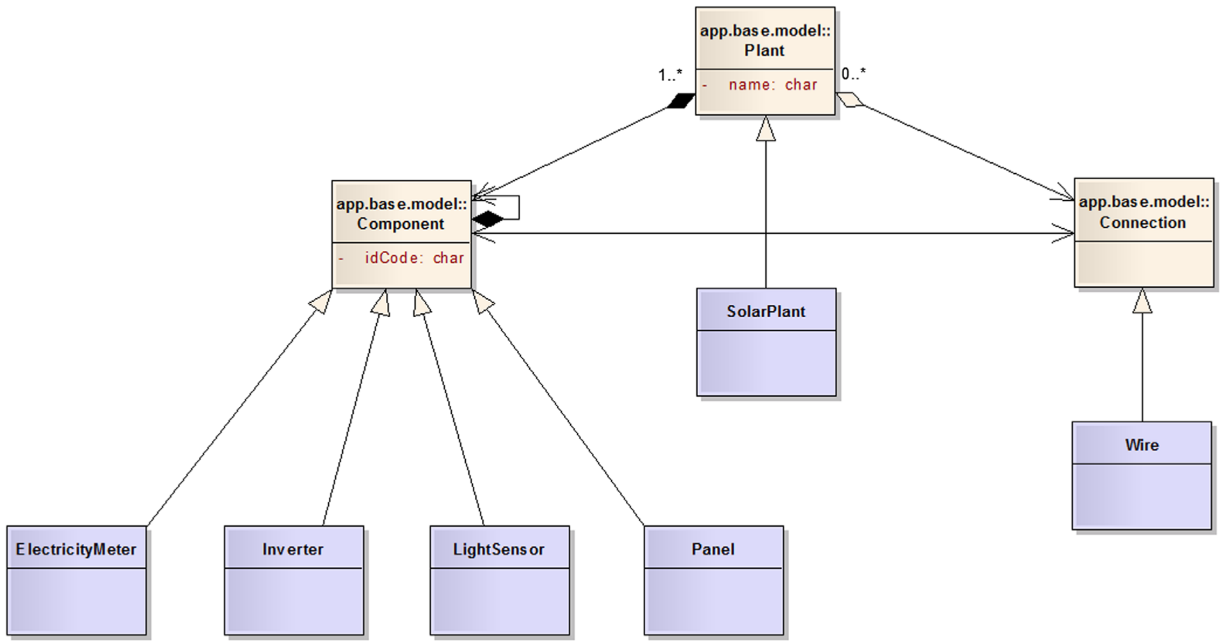


Fig. 3. Photovoltaic plant model.

5. Conclusions

As shown by many authors ([1–7]) monitoring is essential in any renewable energy generation plant. Many of them are focused on specific problems of different types of plants, while in this work the focus is on the very monitoring, describing the necessary parts to achieve it. Ancillary services for DG, such as voltage control [8], will strongly benefit by using information provided by the monitoring systems.

On the other hand, papers like [9–11] enforces our idea of plug-ins as the most appropriate tool for mounting an architecture that can respond to these generic needs of any monitoring system, as well as provide facilities to extend the system with new functionality or new connections to Distributed Resources.

The proposed system is based on 3 major subsystems that cover the basic needs of a generic monitoring system (data collection, visualization of these data and notification of certain states). The entire application design is crossed transversely by the Data Model, which is feeded by the data acquisition module.

The result is a well-modularized system, whose components have specific functions to perform, which can be extended in a customized fashion for particular cases.

6. Future work

To conclude, we present some improvements that can be done on the system to achieve a complete software application:

– ModbusReceiver: based on Modbus [14] protocol a new Receiver might be implemented that would connect, for instance with a Sunny Boy 700 US Inverter [15]. This is an example of an Inverter implemented according to IEE 1547 standard [12].
– URL Receiver: It is possible to implement a configurable Receiver which generates URLs for webservices invocation. As stated in [16] it would be highly useful to access information from any distributed generation electronic device that complies with the standard IEC 61850 [17] through web-services.

References

[1] Lio Kuo-Hua. Dynamic characteristics and graphic monitoring design of photovoltaic energy conversion system. WSEAS Transactions on Systems1109-2777 2011;10(8):239–48.
[2] Wang Li, Liu Kuo-Hua. Implementation of a web-based real-time monitoring and control system for a hybrid wind-pv-battery renewable energy system. Intelligent Systems Applications to Power Systems 2007:1–6.
[3] du Boulay D, Chee C, Chiu K, Leow R, McMullen DF, Quilici R, Turner P. Portal services for collaborative remote instrument control, monitoring and data access, e-science and grid computing. In: Proceedings of IEEE international conference; 2007, p. 328–335, 978-0-7695-3064-2.
[4] Koutroulis Eftichios, Kalaitzakis Kostas. Development of an integrated data-acquisition system for renewable energy sources systems monitoring. Renewable Energy 2003;28(1):139–52 (ISSN: 0960-1481, http://dx.doi.org/10.1016/S0960-1481(01)00197-5).
[5] Forero N, Hernández J, Gordillo G. Development of a monitoring system for a PV solar plant. Energy Conversion and Management 2006;47:15–6 (ISSN: 0196-8904, http://dx.doi.org/10.1016/j.enconman.2005.11.012, p. 2329–2336).
[6] Benghanem M, Maafi A. Data acquisition system for photovoltaic systems performance monitoring, instrumentation and measurement technology conference, IMTC/97'. In: Proceedings of IEEE sensing, processing, networking, vol. 2; May 1997, p. 1030–1033, 0-7803-3747-6.
[7] van Dyk EE, Gxasheka AR, Meyer EL. Monitoring current–voltage characteristics of photovoltaic modules. In: Proceedings of the twenty-ninth IEEE conference record of the photovoltaic specialists; May 2002, 0-7803-7471-1, p. 1516–1519.
[8] Antonio Camacho Miguel, Castilla Jaume, Miret Juan C, Vasquez Eduardo, Alarcón-Gallo. Flexible voltage support control for three phase distributed generation inverters under grid fault. In: Proceedings of IEEE transactions on industrial electronics, 0278-0046.
[9] Wagner S, Winkler S, Pitzer E, Kronberger G, Beham A, Braune R, Affenzeller M. Benefits of plugin-based heuristic optimization software systems. Computer Aided Systems Theory—EUROCAST 2007;4739:747–54 (Copyright: 2007).
[10] Adamou A, Presutti V, Gangemi A. Kali-ma: a semantic guide to browsing and accessing functionalities in plugin-based tools. Knowledge Engineering and Management by the Masses 2010;6317:483–92 (Copyright, 2010).
[11] Chatley R, Eisenbach S, Magee J. Painless plugins, Technical Report: Imperial College London; 2003 ⟨http://www.doc.ic.ac.uk/∼rbc/writings/pp.pdf⟩.

[12] IEEE 1547 Series of Standards, http://grouper.ieee.org/groups/scc21/dr_shared/, 0-7381-5634-5.
[13] World Wide Web Consortium, 〈http://www.w3.org/〉.
[14] Modbus Protocol, 〈http://www.modbus.org/〉.
[15] Sunny Boy 700-US, http://www.sma-america.com/en_US/products/grid-tied-inverters/sunny-boy/sunny-boy-700-us.html.
[16] Pedersen AB, Hauksson EB, Andersen PB, Poulsen B, Træholt C, Gantenbein D. Facilitating a generic communication interface to distributed energy resources. In: Proceedings of the first IEEE international conference on smart grid communications (SmartGridComm); 2010, 978-1-4244-6510-1.
[17] IEC 61850, 〈http://www.iec.ch/,ISBN:978-620-1-61104-7〉.